NAG 1-634

CONCURRENT ALGORITHMS FOR TRANSIENT NONLINEAR FE ANALYSIS

LANGLEY GRANT
IN-61-CR
105439
12P.

by

M. Ortiz

Brown University
Division of Engineering
Providence, RI 02912

1. Introduction

Considerable effort is presently being devoted to developing computational techniques which make efficient use of the new emerging computer architectures. Most methods under investigation at best result in speed-up ratios of $O(p)$ in a p-processor machine. To go beyond this limit, the specific features of the problem under consideration need to be exploited to the fullest extent possible.

Transient problems such as structural dynamics offer ample opportunities in this respect. Here, the property of the solution to be exploited is the fact that information flows between the various parts of the structure at a finite rate rather than instantaneously, as is the case in elliptic problems. Hence, approximations can be introduced concerning the way in which the subdomains interact among themselves. For instance, interactions can be confined to next neighbors, next to next neighbors, etc., with a view to increasing the efficiency of the algorithm.

The present work is concerned with a two-parameter class of time-stepping algorithms for nonlinear structural dynamics possessing the following properties. Let the structure have n degrees of freedom partitioned into s element groups or subdomains. Then:

i) Newmark's method is obtained for the trivial partition ($s = 1$).

ii) The method is unconditional stable for all s within a certain range of the parameters.

iii) Complete concurrency is achieved on a p-processor machine, $p \leq n$, execept for a sequential $O(n)$ operation (mass-averaging).

iv) For a given accuracy of the solution, the speed-up is (asymptotically as $n/s \to \infty$) of $O(p\sqrt{s})$ in 2D and $O(ps)$ in 3D. In particular, speed-ups with respect to Newmark's method ($s = 1$) of $O(\sqrt{s})$ in 2D and $O(s)$ in 3D are obtained on a single processor machine.

In this report, we briefly summarize what facts are presently known about the method and point to directions of current and future research.

2. A Class of Concurrent Algorithms for Nonlinear Structural Dynamics

Throughout this paper we confine our attention to problems in structural dynamics. In the nonlinear case, the equations of motion can be expressed

$$\mathbf{M}\ddot{\mathbf{d}} + \mathbf{F}(\mathbf{d}, \dot{\mathbf{d}}) = \mathbf{f}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{F}$ and $\mathbf{f}$ are the internal and external force vectors and $\mathbf{d}$, $\dot{\mathbf{d}}$ and $\ddot{\mathbf{d}}$ are the displacement, velocity and acceleration vectors, respectively. In the linear case one has

$$\mathbf{F}(\mathbf{d}, \dot{\mathbf{d}}) = \mathbf{K}\mathbf{d} + \mathbf{C}\dot{\mathbf{d}}$$

where $\mathbf{K}$ and $\mathbf{C}$ signify the stiffness and damping matrices, respectively.

Given some set of initial conditions $\mathbf{d}(0) = \mathbf{d}_0$, $\dot{\mathbf{d}}(0) = \mathbf{v}_0$ and the force history $\mathbf{f}(t)$ we wish to integrate the equations of motion incrementally in time. A preliminary step for the application of the method discussed herein is to partition the finite element mesh into subdomains. In a multiprocessing environment it is of primary interest to be able to process the subdomains in parallel. In general, this is a nontrivial proposition in view of the fact that the subdomains are likely to be strongly coupled.

Box 1. A Class of Concurrent Algorithms

- Predictor phase:

$$\tilde{\mathbf{d}}_{n+1} = \mathbf{d}_n + \Delta t \mathbf{v}_n + (1/2 - \beta)\Delta t^2 \mathbf{a}_n$$

$$\tilde{\mathbf{v}}_{n+1} = \mathbf{v}_n + (1 - \gamma)\Delta t \mathbf{a}_n$$

- Equation solving phase:

```
$\mathbf{a}_{n+1} = \mathbf{0}$
    $for\ s = 1, NS\ do$
    $\tilde{\mathbf{a}}^s_{n+1} = -(\mathbf{M}^s + \beta\Delta t^2 \mathbf{K}^s)^{-1}\mathbf{K}^s\tilde{\mathbf{d}}^s_{n+1}$
    $\mathbf{a}_{n+1} = \mathbf{a}_{n+1} + \mathbf{M}^s\tilde{\mathbf{a}}^s_{n+1}$
$\mathbf{a}_{n+1} = \mathbf{M}^{-1}\mathbf{a}_{n+1}$
```

- Corrector phase:

$$\mathbf{d}_{n+1} = \tilde{\mathbf{d}}_{n+1} + \beta\Delta t^2 \mathbf{a}_{n+1}$$

$$\mathbf{v}_{n+1} = \tilde{\mathbf{v}}_{n+1} + \gamma\Delta t \mathbf{a}_{n+1}$$

A time-stepping algorithm which circumvents this difficulty is shown in Box 1, where the linear, undamped, unforced case is considered for simplicity. Extensions to the general case are straightforward. The algorithm comprises three phases. The predictor and corrector steps are identical to those in Newmark's method. However, the equation solving phase is designed to introduce the desired degree of parallelism into the computations. First, the predictor displacements $\tilde{\mathbf{d}}_{n=1}$ are localized into the subdomains to obtain a collection of local predictors $\{\tilde{\mathbf{d}}^s_{n+1},\ s = 1, \ldots, NS\}$. The corresponding local acceleration arrays $\tilde{\mathbf{a}}^s_{n+1}$ are then computed from the local predictors $\mathbf{d}^s_{n+1}$ by applying Newmark's update at the subdomain level. During this operation the subdomains are regarded as being decoupled from each other. The local acceleration arrays so obtained are generally multivalued at the subdomain boundaries. Compatibility between the subdomains is restored using a mass averaging rule

$$\mathbf{a}_{n+1} = \mathbf{M}^{-1} \sum_{s=1}^{NS} \mathbf{M}^s \tilde{\mathbf{a}}^s_{n+1}$$

which completes one application of the algorithm. In the nonlinear case the local updates take the form

$$\mathbf{M}^s \tilde{\mathbf{a}}^s_{n+1} + \mathbf{F}^s(\tilde{\mathbf{d}}^s_{n+1} + \beta \Delta t^2 \tilde{\mathbf{a}}^s_{n+1}) = \mathbf{0}$$

which defines a set of local systems of nonlinear equations which need to be solved for the local acceleration predictors $\tilde{\mathbf{a}}^s_{n+1}$. This can be accomplished by means of a local Newton-Raphson iteration or some other nonlinear solution procedure.

The choice of averaging rule to restore compatibility of accelerations at the subdomain boundaries is not arbitrary. It has been shown [1] that the mass averaging rule is the only choice which results in consistency with the equations of motion. It has also been shown in [1] that the stability properties of the method are identical to those of Newmark's method regardless of the choice of mesh partition. Thus, the algorithm results in an oscillatory, unconditionally stable response for $\gamma \geq 1/2$, $\beta \geq \gamma/2$.

Some particular cases of the proposed method are noteworthy. Thus, for the trivial partition, i. e., that obtained from considering one single subdomain coincident with the total structure, Newmark's method is recovered. Assuming that the properties of the algorithm depend continuously on the number of subdomains, the performance of the method can be expected to be close to that of Newmark's scheme for a small number of subdomains and to gradually depart from it as the number of subdomains is increased. In the limit of an element-by-element partition in which the subdomains are identified with the elements in the mesh, the algorithm takes an entirely explicit character with all factorizations being performed at the element level.

Since the subdomains are decoupled during the equation solving phase, all subdomains can be processed in parallel. Furthermore, the equation solving effort is reduced to factorizing the local amplification matrices. No global array needs to be formed, much less factorized. Combined with the parallelism in the computations, the local character of the matrix factorizations results in significant speed-up factors compared to globally implicit methods.

3. Computational Efficiency

To estimate the computational efficiency of the method, let us start by recalling that the number of operations involved in matrix factorization and forward and backward substitution is

$$FACTORIZATION \approx \frac{1}{2}nb^2, \qquad SUBSTITUTION \approx 2nb$$

where b is the semiband width and n, as before, is the number of degrees of freedom of the structure. The cost of large scale nonlinear computations is dominated by the equation solving phase. Under these conditions, a good estimate of the computational cost is given by

$$COST \approx (FACTORIZATION + SUBSTITUTITON) \times ITERATIONS \times STEPS$$

where $ITERATIONS$ is the average number of equilibrium iterations per time step and $STEPS$ is the number of time steps required for a given duration of the analysis T, i. e., $STEPS = T/\Delta t$.

In 2D, consider a square mesh of l^2 elements. Then, $b = l+2$, $n = (l+1)^2$ and, thus, a global system solution requires

$$GLOBAL \approx \frac{1}{2}(l+2)^2(l+1)^2 + 2(l+2)(l+1)^2$$

operations. Assume now that the mesh is partitioned into $s = m^2$ subdomains. Then, the equation solving effort involved in one application of the partitioned algorithm is

$$PARTITIONED \approx s\left[\frac{1}{2}\left(\frac{l}{m}+2\right)^2\left(\frac{l}{m}+1\right)^2 + 2\left(\frac{l}{m}+2\right)\left(\frac{l}{m}+1\right)^2\right]$$

For nontrivial partitions, this count is less than that pertaining to the global system. Thus, the net speed-up in equation solving afforded by the partitioning is given by

$$SPEED-UP = \frac{GLOBAL}{PARTITIONED}$$

The dependence of this function on the number of subdomains is shown in Fig. . It is readily verified that a speed-up of order s is attained asymptotically in the large scale limit $n/s \rightarrow \infty$.

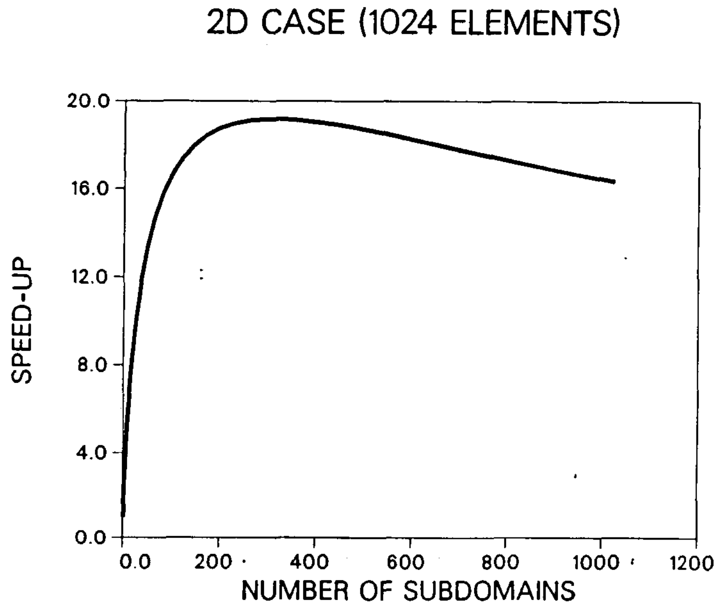

Fig. 1. Speed-up of equation solving computations on a single processor as the mesh is partitioned into an increasing number of subdomains. Two dimensional case.

The 3D case is amenable to an entirely similar analysis. The resulting speed-up is shown in Fig. 2 as a function of the number of subdomains. Here, an asymptotic speed-up of order of $s^{4/3}$ is reached in the large scale limit.

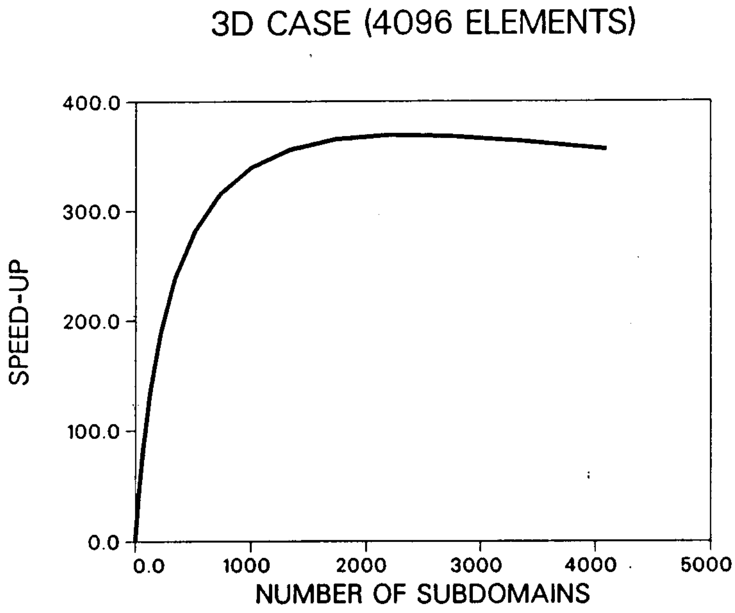

Fig. 2. Speed-up of equation solving computations on a single processor as the mesh is partitioned into an increasing number of subdomains. Three dimensional case.

Some aspects of these estimates are noteworthy. Firstly, it is seen that some efficiency is gradually lost for a given size n as the number of subdomains s is increased. This loss is due to the fact that the interface nodes need to be reduced more than once during the subdomain factorizations. On the other hand, it should be noted that these speed-ups cannot be fully realized in practice due to the fact that, in order to maintain the accuracy of the solution, the time step needs to be cut down as the number of subdomains is increased.

It turns out, however, that accuracy constraints offset the factorization speed-ups only partially and net gains remain. To see this, we need to estimate the time-steps required to maintain a prespecified level of accuracy as the number of subdomains is increased. In [2] this was accomplished by adapting an analysis of Mullen and Belytcshko [3] to the present situation. The main conclusion is that the time step needs to be reduced as $O(1/s^{1/2})$ in 2D and as $O(1/s^{1/3})$ in 3D. This leaves a net speed-up of $O(s^{1/2})$ in 2D and $O(s)$ in 3D, which in conjunction with the $O(p)$ speed-up afforded by concurrency yields

$$SPEED - UP(2D) = O(p\sqrt{s}), \qquad SPEED - UP(3D) = O(ps)$$

Here, instead, we wish to confirm these estimates by way of numerical testing. To this end, we choose the problem of a square membrane undergoing large deflections. The membrane is supported all around and subjected to a uniform initial velocity throughout its interior. The magnitude of the initial velocity is substantial enough to generate strains of the order of 30% and rotations of the order of 45°.

The element utilized in the calculations is a four node quadrilateral obtained by averaging two triangular assemblies, corresponding to the common side of the triangles being aligned with each one of the diagonals. The constituent triangular elements are endowed with a strain energy of the form

$$W = \frac{T}{2}\frac{A^2}{A_0}$$

where T is the tension of the membrane, and A and A_0 are the areas of the deformed and underformed triangles. It is easily checked that this formulation reduces to the usual small deflection theory of membranes when $A \approx A_0$.

Fig. 3 shows the computed center deflections for various partitions of the mesh. The values of the material parameters adopted were $T = 1$ and a mass density $\rho = 1$. The half size of the membrane was taken to be $L = 1$. By virtue of the symmetries of the problem, only one quarter of the membrane needs to be discretized. The mesh used in the calculations comprised 1024 identical four noded elements.

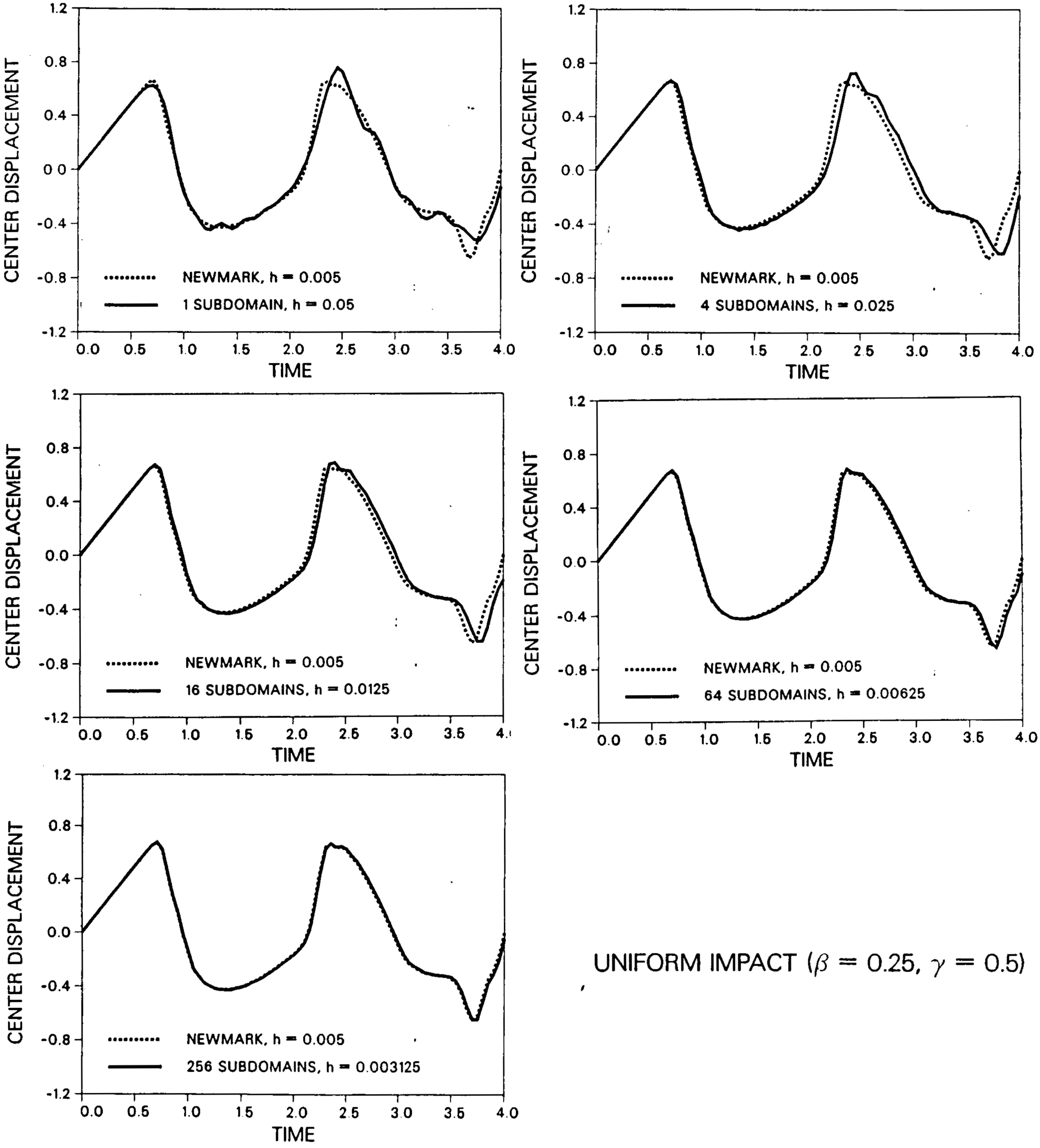

Fig. 3. Computed center deflection histories for two dimensional large deformation membrane problem.

The calculations are carried out for various time steps around the theoretical estimate derived in [2]. The error in the solution is then computed as

$$ERROR^2 = \int_0^T | w(t) - w_{exact}(t) |^2 \frac{dt}{t^2}$$

where $w(t)$ and $w_{exact}(t)$ are the computed and exact center deflections, respectively. In lieu of an exact solution, the results from Newmark's method with a small time step ($\Delta t = 0.005$) are utilized. The above definition of the error provides a measure of the period elongation in the computed solution. In particular, it can be shown that

$$\lim_{T\to\infty} \left[\int_0^T | sin(\omega t) - sin\big((\omega + \Delta\omega)t\big) |^2 \frac{dt}{t^2} \right]^{1/2} \propto \Delta\omega$$

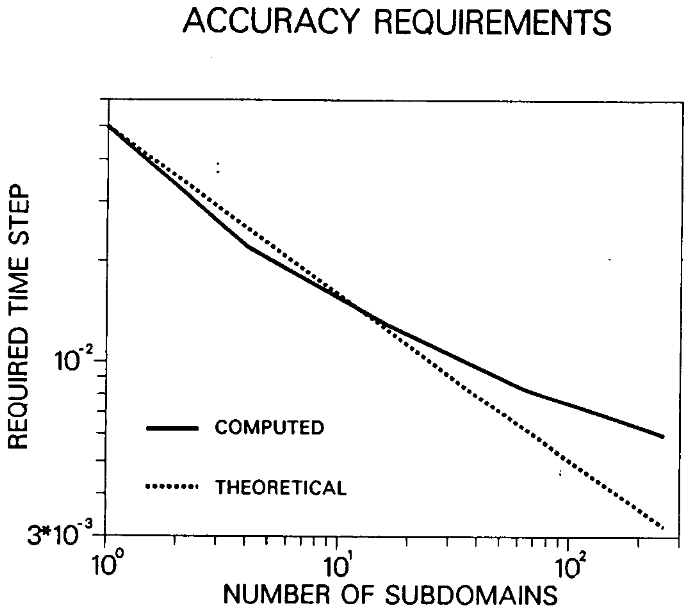

Fig. 4. Time steps required to preserve the level of accuracy in the solution as the number of subdomains is increased.

These data were then utilized to pinpoint the time steps required to maintain a level of accuracy equal to that of Newmark's method with $\Delta t = 0.05$. The results are shown in Fig. 4, together with the theoretical estimate derived in [2]. As may be seen, the theoretical accuracy requirements are realized quite closely.

TABLE 1.- Equation solving timings on one processor.

Membrane example.

1024 ELEMENT CASE			
NSUB	Secs.	Speed-up	Theory
1	1143	1	1
4	776	1.47	2
16	521	2.19	4
64	326	3.51	8
256	156	7.31	16

Finally, Table 1 records the computed equation solving cost as a function of the number of subdomains. This dependency is the combined effect of the slow-down due to accuracy loss shown in Fig. 4 and the speed-up due to factorization and substitution savings depicted in Fig. 1. Also shown is the theoretical $O(\sqrt{s})$ speed-up. It is apparent from these results that the theoretical estimate is indeed asymptotic and is only realized in the large scale limit $n/s \rightarrow \infty$. For a structure the size of the one tested, the net speed-ups obtained are a fraction of the asymptotic predictions, in spite of which the gains are rather substantial. For instance, for 256 subdomains a net seven fold speed-up is obtained over Newmark's solution.

4. Summary and Present Directions of Progress

What sets the present method apart from other concurrent algorithms is the fact that it can be used to some advantage in sequential machines as well. Thus, substantial speed-ups are obtained on a single processor as the number of subdomains is increased. An additional $O(p)$ speed-up is obtained when p processors are utilized.

Present work is proceeding in several directions. The test case discussed above is being repeated for a mesh comprising four times as many elements (4096), in an effort to understand how the large scale asymptotic speed-ups are attained. A three dimensional example involving finite deformations and free body motions is also being pursued. A code optimized for concurrency in the Alliant FX8 computer is being finalized. This will provide the means for testing the performance of the algorithm in a multiprocessor environment. Future plans call for running similar tests on our in-house 32 node Intel hypercube.

References

1. M. Ortiz and B. Nour-Omid, 'Unconditionally stable concurrent procedures for transient finite element analysis,' *Comp. Meth. Appl. Mech. Engng.*, **58**, 151-174 (1986).
2. M. Ortiz, B. Nour-Omid and E. D. Sotelino, "Accuracy of a Class of Concurrent Algorithms for Transient Finite Element Analysis," *Int. J. Num. Meth. Engr.*, (to appear).
3. R. Mullen and T. Belytchko, 'An analysis of an unconditionally stable explicit method, *Computers and Structures*, **16**, 691-696 (1983).